

iAPX 86,88 FAMILY UTILITIES USER'S GUIDE

Order Number: 121616-003

Additional copies of this manual or other Intel literature may be obtained from:

Literature Department
Intel Corporation
3065 Bowers Avenue
Santa Clara, CA 95051

The information in this document is subject to change without notice.

Intel Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Intel Corporation assumes no responsibility for any errors that may appear in this document. Intel Corporation makes no commitment to update nor to keep current the information contained in this document.

Intel Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in an Intel product. No other circuit patent licenses are implied.

Intel software products are copyrighted by and shall remain the property of Intel Corporation. Use, duplication or disclosure is subject to restrictions stated in Intel's software license, or as defined in ASPR 7-104.9(a)(9).

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Intel Corporation.

The following are trademarks of Intel Corporation and its affiliates and may be used to identify Intel products:

BXP	Intel	iSBX	Multichannel
CREDIT	Int ₂ l	Library Manager	Multimodule
i	Intelevison	MCS	Plug-A-Bubble
ICE	Inteltec	Megachassis	PROMPT
iCS	iOSP	Micromainframe	RMX/80
i _m	iPDS	Micromap	System 2000
iMMX	iRMX	Multibus	UPI
Insite	iSBC		

REV.	REVISION HISTORY	DATE
-001	Original issue.	9/80
-002	Adds information to support V2.0 of LINK86, LOC86, and LIB86 and V1.0 of CREF86; corrects technical and typographical errors.	11/81
-003	Adds examples for iRMX 86 operating environment.	5/82





This manual describes how to use the iAPX 86,88 Family utilities:

- LINK86
- CREF86
- LIB86
- LOC86
- OH86

These products run on 8086- and 8088-based systems. They are used by programmers developing programs with ASM86, ASM89, PL/M-86, PASCAL-86, FORTRAN-86, or any other language translator that produces object code compatible with the iAPX 86,88 Family of processors. The iAPX 86,88 Family of processors includes 8086, 8088, 8087, and 8089 processor chips. Because the 8086 is the first member of this family, this manual uses 8086 generically to represent the entire family.

This manual presumes familiarity with the conventions of the operating system under which the iAPX 86,88 utilities are being executed. It also presumes familiarity with the basic requirements of individual languages and translators.

This manual is divided into the following chapters:

- Chapter 1, Introduction: a summary of the relationship among the utilities and basic concepts governing their use
- Chapter 2, LINK86: how to invoke, use the controls for, and read the printed listing from LINK86
- Chapter 3, CREF86: how to invoke, use the controls for, and read the output listing from CREF86
- Chapter 4, LIB86: how to invoke and use the commands for LIB86
- Chapter 5, LOC86: how to invoke, use the controls for, and read the printed listing from LOC86
- Chapter 6, OH86: how to invoke OH86

This manual also contains several appendixes, meant for quick access to the following information:

- iAPX 86,88 absolute object file format definitions (Appendix A)
- Hexadecimal-decimal conversion information (Appendix B)
- The effect of available memory on the performance of LINK86, CREF86, LIB86, and LOC86 (Appendix C)
- Summaries of iAPX 86,88 Family utility controls and error messages:
 - LINK86 (Appendix D)
 - CREF86 (Appendix E)
 - LIB86 (Appendix F)
 - LOC86 (Appendix G)
 - OH86 (Appendix H)

NOTE

The following appendixes address issues dependent on specific operating systems, such as operating environments, related publications, and examples. These appendixes contain foldout pages, designed to be opened out to your right and used in conjunction with general instructions provided in the chapters and other appendixes. On these foldout pages you will find sample invocations for the iAPX 86,88 Family utility controls and commands.

- Additional information for Series III users (Appendix I)
- Additional information for iRMX 86 users (Appendix J)

Once you have gained sufficient familiarity with the basic principles of iAPX 86,88 Family utilities operation, you will find the following publication convenient for quick syntax reference:

- *iAPX 86,88 Family Utilities Pocket Reference*, order number 121669.

Before reading this manual, ensure that you are familiar with the following terms and conventions:

Notational Conventions

punctuation	other than the following must be entered if required by the control syntax.
{ }	indicates that one and only one of the syntactic items contained within the braces is required.
[]	indicates that the syntactic item or items contained within the brackets are optional.
...	indicates that the preceding syntactic item may be repeated an indefinite number of times. (The ellipsis is often used within brackets and with a comma "[,...]" to indicate that preceding item may be repeated, but each repetition must be separated by a comma.)
	separates various options within the brackets [] or braces { }.
UPPERCASE	indicates that these characters must be entered exactly as shown.
<i>italic</i>	indicates a meta symbol that may be replaced with an item that fulfills the rules for that symbol. The actual symbol may be any of the following:
<i>pathname</i>	is a valid designation for a file; in its entirety, it consists of a <i>directory-name</i> and a <i>filename</i> .
<i>directory-name</i>	is that portion of a <i>pathname</i> that acts as a file locator by identifying the device and/or directory containing the <i>filename</i> .
<i>filename</i>	is a valid name for the part of a <i>pathname</i> that names a file.

<i>minimum-size</i> <i>maximum-size</i> <i>paragraph</i> <i>offset</i> <i>address</i>	are numbers and must follow Intel standards for number representation (see PL/M-86 or ASM86). Use the H suffix for hexadecimal, B suffix for binary, O or Q suffix for octal and D or nothing for decimal.
<i>segment name</i> <i>module name</i> <i>class name</i> <i>group name</i> <i>overlay name</i> <i>public symbol</i> <i>variable name</i>	are defined by the 8086 object file formats described in Appendix A. They may be up to forty characters long and may contain any of the following characters in any order: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ?, @, ., :, ,, —,
black	black background is used in examples to indicate the user's entries.
<i>system-id</i>	is a generic label placed on sample listings where an operating system-dependent name would actually be printed.
<i>pathname1,</i> <i>pathname2, ...</i>	are generic labels placed on sample listings where one or more user-specified pathnames would actually be printed.
<i>Vx.y</i>	is a generic label placed on sample listings where the version number of the product that produced the listing would actually be printed.





ILLUSTRATIONS

FIGURE	TITLE	PAGE	FIGURE	TITLE	PAGE
1-1	The iAPX 86,88 Family Development Process	1-1	6-1	OH86 Input and Output Files	6-1
1-2	Library Linkage by LINK86	1-3	I-1	LINK86 Print File for Bound Object Module	I-3
1-3	The LINK86/LOC86 Process	1-4	I-2	LOC86 Print File for Bound Object Module	I-4
1-4	8086 Addressing	1-6	I-3	LINK86 Default Print File	I-5
1-5	Segment Physical Relationships	1-7	I-4	CREF86 Cross-Reference Listing	I-6
1-6	Segment Alignment Boundaries	1-8	I-5	LINK86 Listing for Program with Overlays	I-9
1-7	Memory Configuration of Program with Overlays	1-10	I-6	LOC86 Listing for Program with Overlays	I-10
2-1	LINK86 Input and Output Files	2-1	I-7	LINK86 Map for Root File	I-11
2-2	LINK86 Print File Header	2-27	I-8	Module Information for Overlays	I-12
2-3	LINK86 Link Map	2-27	I-9	Memory Organization for Example 6	I-13
2-4	LINK86 Group Map	2-28	J-1	CREF86 Cross-Reference Listing	J-4
2-5	LINK86 Symbol Table	2-29	J-2	LINK86 Listing for Program with Overlays	J-7
3-1	CREF86 Input and Output Files	3-1	J-3	LOC86 Listing for Program with Overlays	J-8
3-2	Header of Cross-Reference Listing	3-7	J-4	LINK86 Map for Root File	J-9
3-3	Warning Messages on CREF86 Listing	3-7	J-5	Module Information for Overlays	J-10
3-4	Module List on CREF86 Listing	3-8	J-6	Memory Organization for Example 4	J-11
3-5	Symbol Cross-Reference Information	3-9			
5-1	LOC86 Input and Output Files	5-1			
5-2	LOC86 Symbol Table	5-22			
5-3	LOC86 Memory Map	5-23			
5-4	LOC86's Address Assignments for Overlays	5-26			



Generating Code to Run on an iRMX 86-Based System

To generate code that runs on an iRMX 86-based system, perform the following steps:

1. Translate the program into object code by using the appropriate compiler or assembler.
2. Use LINK86 to link the program with other routines or libraries as necessary. When doing this, remember the following:
 - If you wrote your program in FORTRAN or Pascal, or if you invoked specific universal development interface (UDI) calls, you must link your program to the iRMX 86 UDI library that corresponds to the model of segmentation for your program. These libraries are:

Library	Model of Segmentation
URXLRG.LIB	LARGE or MEDIUM
URXCOM.LIB	COMPACT
URXSML.LIB	SMALL

- Do not use FASTLOAD control. Currently, the iRMX 86 Operating System cannot load programs linked with this control.
 - To produce LTL code, use the BIND control. In this case, also specify the MEMPOOL and SEGSIZE controls to allocate memory for the memory pool and stack. If you do not use BIND, you must specify SEGSIZE with the LOC86 command.
3. If you did not specify the BIND control in the LINK86 command, use LOC86 to assign absolute addresses to your program. In order to run this program in an iRMX 86 environment, you must also reserve the program's memory locations during iRMX 86 configuration.
 4. To invoke the program from a terminal, enter the pathname of the file that contains the program's linked (if LTL code) or located object code.

Program Development Examples

The following examples are programming problems solved by using one or more of the iAPX 86,88 utilities on an iRMX 86-based system.

Example 1: Using CREF86

Figure J-1 illustrates a CREF86 cross-reference listing for an input list of 15 files, one of which contains several modules. The output print file pathname OUT and a title for the listing were specified in the controls. Although PAGEWIDTH (PW) and PAGELength (PL) specifications were also noted in the controls, the numbers specified are the same as those provided by default.

```

CREF86      EXAMPLE OF CROSS REFERENCE USING CREF86
MM/DD/YY
PAGE      1

IRMX 86 CREF86 Vx.y

INPUT FILES:      :F1:ROOT.OBJ      :F1:PARSE.OBJ      :F1:SIGNON.OBJ      :F1:STATE.OBJ      :F1:ERROR.OBJ      :F1:UTILS.OBJ
                  :F1:HEMMAH.OBJ     :F1:SCANNR.OBJ     :F1:PROCES.OBJ      :F1:SCUTIL.OBJ     :F1:LIST.OBJ       :F1:LSUTIL.OBJ
                  :F1:SORT.OBJ       :F1:UDSMA.LNK      URXCON.LIB

OUTPUT FILE:      OUT
CONTROLS SPECIFIED: PR(OUT) TT(EXAMPLE OF CROSS REFERENCE USING CREF86) PW(120) PL(60)

MODULES INCLUDED:

FILE NAME      MODULE NAME(S)

:F1:ROOT.OBJ:      CREF86
:F1:PARSE.OBJ:     PARSE
:F1:SIGNON.OBJ:    SIGNON
:F1:STATE.OBJ:     NEXTSTATE
:F1:ERROR.OBJ:     ERROR
:F1:UTILS.OBJ:     UTILITIES
:F1:HEMMAH.OBJ:    MEMORYMANAGEMENT
:F1:SCANNR.OBJ:    SCANMODULES
:F1:PROCES.OBJ:    PROCESSRECORDS
:F1:SCUTIL.OBJ:    SCANUTILITIES
:F1:LIST.OBJ:      LISTOUTPUT
:F1:LSUTIL.OBJ:    LISTUTILITIES
:F1:SORT.OBJ:      SYMBOLSORT
:F1:UDSMA.LNK:     OBJMAN
URXCON.LIB        DQALLOCATE      DQATTACH      DQCHANGEEXTENSION  DQCREATE      DQCODESEXCEPTION
                  DQEXIT        DQFREE        DQGETARGUMENT      DQGETSYSTEMID
                  DQGETTIME     DQOPEN        DQREAD             DQSEEK        DQWRITE
                  SYSTEMSTACK

```

CREF86		EXAMPLE OF CROSS REFERENCE USING CREF86		MM/DD YY		PAGE 2	
SYMBOL NAME		SYMBOL TYPE		DEFINING MODULE; REFERRING MODULE(S)			
-----		-----		-----			
ACCESS_PAGE		UNKNOWN		OBJMAN			
ALLOCATE		UNKNOWN		OBJMAN			
APPENDNODE		PROCEDURE NEAR		UTILITIES			
APPENDUDSMNODE		PROCEDURE NEAR		UTILITIES; PARSE SCANMODULES PROCESSRECORDS			
ARRAYBASE		POINTER		SYMBOLSORT; LISTOUTPUT			
ATOI		PROCEDURE WORD NEAR		UTILITIES; PARSE			
BTOX		PROCEDURE WORD NEAR		UTILITIES; LISTUTILITIES			
BUBBLESORTVARNAMES		PROCEDURE NEAR		SYMBOLSORT; LISTOUTPUT			
BUMPLINECOUNT		PROCEDURE NEAR		LISTUTILITIES; LISTOUTPUT			
CHECKHEADER		PROCEDURE NEAR		SCANUTILITIES; SCANMODULES			
CHECKOVERLAY		PROCEDURE NEAR		SCANUTILITIES; SCANMODULES			
CHECKVARIABLE		PROCEDURE SITE NEAR		SCANUTILITIES; PROCESSRECORDS			

CREF86 EXAMPLE OF CROSS REFERENCE USING CREF86		MM/DD/YY		PAGE 6
UNLOAD_PAGE	UNKNOWN	OBJMAN		
VARNAMEP	POINTER	MEMORYMANAGEMENT;	PROCESSRECORDS	LISTOUTPUT SYMBOLSORT
VBLOCKLISTHEADER	WORD	PROCESSRECORDS;	UTILITIES LISTOUTPUT	SYMBOLSORT
WARNING	PROCEDURE NEAR	ERROR;	SCANMODULES	PROCESSRECORDS
WRITEDATA	PROCEDURE NEAR	LISTUTILITIES;	ERROR UTILITIES	LISTOUTPUT1
WRITEINITLINEBUF	PROCEDURE NEAR	LISTUTILITIES;	ERROR UTILITIES	LISTOUTPUT
WRITELINE	PROCEDURE NEAR	LISTUTILITIES;	LISTOUTPUT	
WRITENEWLINE	PROCEDURE NEAR	LISTUTILITIES;	UTILITIES LISTOUTPUT	
WRITETOCOMMANDBUF	PROCEDURE NEAR	PARSE		
WRITETOFILE	PROCEDURE NEAR	LISTUTILITIES;	ERROR LISTOUTPUT	
ZERO	WORD	UTILITIES		

Figure J-1. CREF86 Cross-Reference Listing

Example 2: Building and Using Library Files

A library is a file that contains object modules. Libraries allow you to collect commonly-used pieces of software into one file. The library file can be included in a LINK86 invocation, and LINK86 will use the modules to resolve references.

You can add the output from a translator, LINK86, or LOC86 to a library. The modules added may be relocatable or absolute; they can have unresolved references or be completely linked.

Let's consider the following scenario—we have created six routines (SINE, COSINE, TANGENT, COSECANT, SECANT, and COTANGENT). We want to create a library file that will allow each routine to be linked to programs separately.

The first step necessary to create the library is to translate each routine separately. If we were to put them in a single source module, the translator would translate them into one module with six public symbols. We could add this module to a library, but when we tried to link one of the routines into a program, all six would be included.

Once the routines are translated, LIB86 can be used to create a library file and add modules. The LIST command is used to display the contents of the library and the publics contained within it.

```
-LIB86<cr>
iRMX 86 8086 LIBRARIAN Vx.y
*CREATE :PROG:LIBRARY/TRIG.LIB<cr>
*ADD :FDO:SIN.OBJ, :FDO:COS.OBJ TO :PROG:LIBRARY/TRIG.LIB<cr>
*LIST :PROG:LIBRARY/TRIG.LIB PUBLICS<cr>

:PROG:LIBRARY/TRIG.LIB
SIN
SINE
COS
COSINE

*ADD :FDO:SEC.LNK, :FDO:CSC.LNK, :FDO:COT.LNK, &<cr>
**FDO:TAN.LNK TO :PROG:LIBRARY/TRIG.LIB <cr>
*LIST :PROG:LIBRARY/TRIG.LIB PUBLICS<cr>

:PROG:LIBRARY/TRIG.LIB
SIN
SINE
COS
COSINE
SEC
SECANT
CSC
COSECANT
COT
COTANGENT
TAN
TANGENT

*EXIT<cr>
```

Example 3: Linking and Locating Programs with Overlays Using OVERLAY Control

The easiest way to build an 8086 program that contains overlays is with LINK86's OVERLAY control. Overlay modules built with this control reside in the same file as the root. The operating system supplies routines that will load the overlays constructed in this way. See the *iRMX 86 Loader Reference Manual* or the *Run-Time Support Manual for iAPX 86,88 Applications*.

After the program modules that will constitute the root and its overlays are translated, each of the overlays and the root must be linked separately. Then the root and all of the overlays are linked together.

The example following shows the first step toward linking overlays—linking all of the modules that will constitute each overlay and the root separately:

```
LINK86 OV1.OBJ, OV1A.OBJ, OV1B.OBJ &<cr>
**OVERLAY(OVERLAY1)<cr>

LINK86 OV2.OBJ, OV2B.OBJ, OV2C.OBJ &<cr>
**OVERLAY(OVERLAY2)<cr>

LINK86 OV3.OBJ, OV3A.OBJ OVERLAY(OVERLAY3)<cr>

LINK86 OV4.OBJ, OV4A.OBJ OVERLAY(OVERLAY4)<cr>

LINK86 ROOT.OBJ, ROOTA.OBJ, ROOTB.OBJ, &<cr>
**URXSML.LIB OVERLAY(ROOT)<cr>
```

Notice that all of the modules, including the root, are linked with the OVERLAY and NOBIND controls. The overlay name for the root is not as critical as for the overlays, since the overlay name is used when calling the loader.

Finally, the overlays and root must be linked together. Since any one of the files could be the root, LINK86 requires for the final link the file containing the root must be first in the input list. During this final link, the OVERLAY control is not used:

```
LINK86 ROOT.LNK, OV1.LNK, OV2.LNK, OV3.LNK, &<cr>
**OV4.LNK TO PROG.86 BIND<cr>
```

In the invocation, the BIND control is specified. The resulting object file is executable on an iRMX 86-based system.

Figure J-2 shows the LINK86 print file listing for the previous invocation.

There is nothing special about the invocation line to LOC86 when locating a file that contains overlays or that has been bound:

```
LOC86 PROG.86 RESERVE (0H TO 77FFH, &<cr>
**0FC000H TO 0FFFFFH)<cr>
```

The RESERVE control prevents LOC86 from assigning memory addresses reserved for the Operating System. However, the values you enter with the RESERVE control must depend on the size and location of your Operating System and other application software. Figure J-3 illustrates the printout from this invocation.

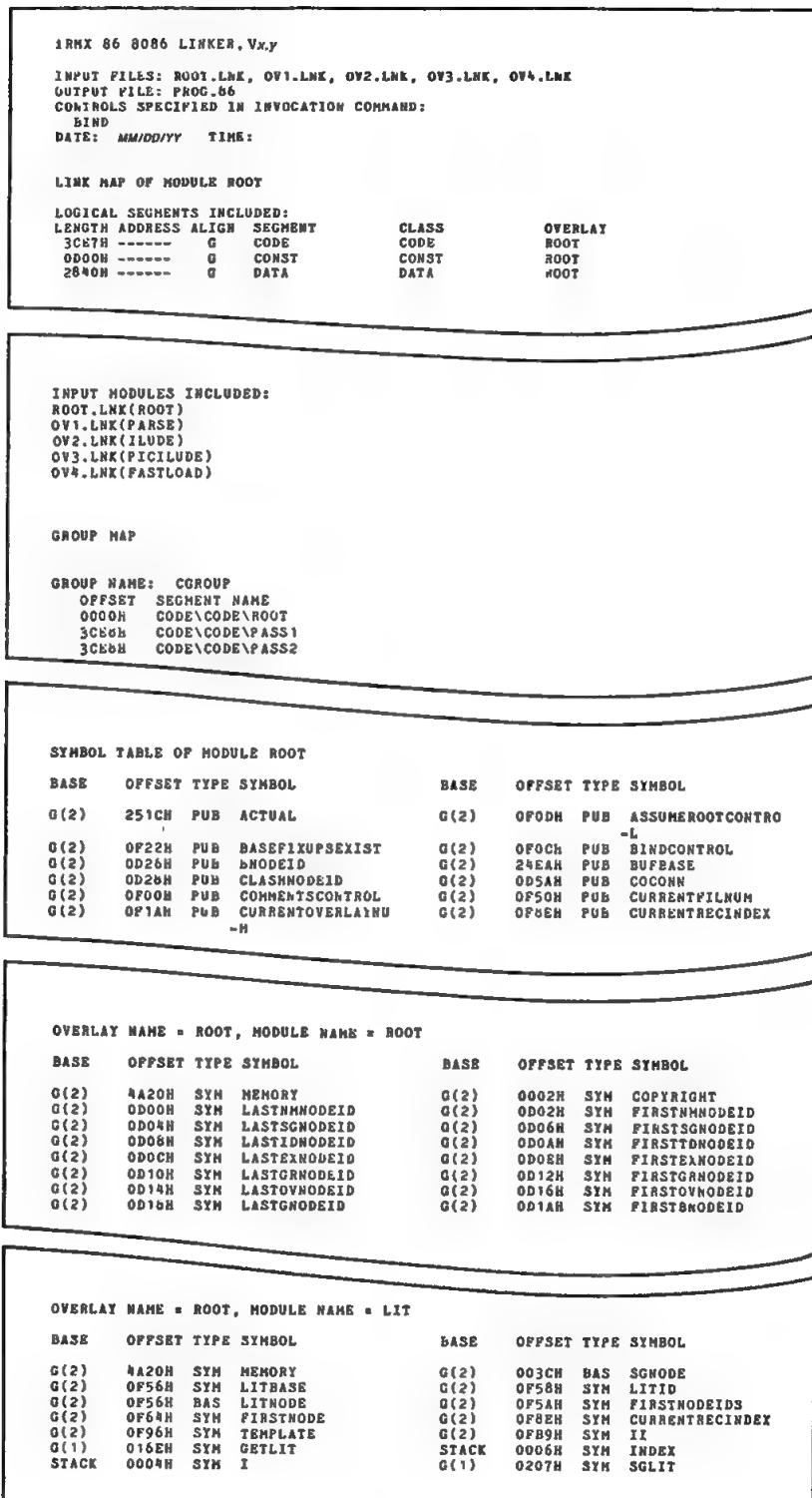


Figure J-2. LINK86 Listing for Program with Overlays

iRMX 86 8086 LOCATOR, Vx.y

INPUT FILE: PROG.b6
 OUTPUT FILE: PROG
 CONTROLS SPECIFIED IN INVOCATION COMMAND:
 RESERVE(0H TO 77FFH,0FC000H TO 0FFFFH)
 DATE: MM/DD/YY TIME:

SYMBOL TABLE OF MODULE ROOT

BASE	OFFSET	TYPE	SYMBOL	BASE	OFFSET	TYPE	SYMBOL
1034H	251CH	PUB	ACTUAL	1034H	0F0DH	PUB	ASSUMEROOTCONTRO -L
1034H	0F22H	PJB	BASEFIXUPSEXIST	1034H	0F0CH	PUB	BINDCONTROL
1034H	0D26H	PUB	BNODEID	1034H	24EAH	PUB	BUFBASE
1034H	0D28H	PUB	CLASHNODEID	1034H	0D5AH	PUB	COCOH
1034H	0F00H	PUB	COMMENTSCONTROL	1034H	0F50H	PUB	CURRENTFILNUM
1034H	0F1AH	PUB	CURRENTOVERLAYNU -M	1034H	0F8EH	PUB	CURRENTRECINDEX

OVERLAY = ROOT, MODULE = ROOT

BASE	OFFSET	TYPE	SYMBOL	BASE	OFFSET	TYPE	SYMBOL
1034H	4A20H	SYM	MEMORY	1034H	0002H	SYM	COPYRIGHT
1034H	0D00H	SYM	LASTHNNODEID	1034H	0002H	SYM	FIRSTHNNODEID
1034H	0D04H	SYM	LASTSGNODEID	1034H	0D06H	SYM	FIRSTSGNODEID
1034H	0D08H	SYM	LASTTDNODEID	1034H	0D0AH	SYM	FIRSTTDNODEID
1034H	0D0CH	SYM	LASTENODEID	1034H	0D0EH	SYM	FIRSTENODEID
1034H	0D10H	SYM	LASTOVNODEID	1034H	0D12H	SYM	FIRSTOVNODEID
1034H	0D14H	SYM	LASTBNODEID	1034H	0D16H	SYM	FIRSTOVNODEID
1034H	0D18H	SYM	LASTBNODEID	1034H	0D1AH	SYM	FIRSTBNODEID
1034H	0D1CH	SYM	SGNODEID	1034H	0D1EH	SYM	GRNODEID

OVERLAY = ROOT, MODULE = LIT

BASE	OFFSET	TYPE	SYMBOL	BASE	OFFSET	TYPE	SYMBOL
1034H	4A20H	SYM	MEMORY	1034H	003CH	SYM	SGNODE
1034H	0F56H	SYM	LITBASE	1034H	0F58H	SYM	LITID
1034H	0F56H	SYM	LITNODE	1034H	0F5AH	SYM	FIRSTNODEIDS
1034H	0F64H	SYM	FIRSTNODE	1034H	0F6EH	SYM	CURRENTRECINDEX
1034H	0F96H	SYM	TEMPLATE	1034H	0F89H	SYM	LI
0780H	0105H	SYM	GETLIT	STACK	0006H	SYM	INDEX
STACK	0004H	SYM	I	0780H	0207H	SYM	SGLI1

MEMORY MAP OF MODULE ROOT

MODULE START ADDRESS PARAGRAPH = 14D6H OFFSET = 0006H
 SEGMENT MAP

START	STOP	LENGTH	ALIGN	NAME	CLASS	OVERLAY
07800H	0B4E6H	3CE7H	H	CODE	CODE	ROOT
0B4E6H	0F9BAH	44D3H	H	CODE	CODE	PASS1
0B4E6H	0E0CEH	2BETH	H	CODE	CODE	PASS2
0B4E6H	10337H	4E50H	H	CODE	CODE	PIC_PASS2

GROUP MAP

ADDRESS	GROUP OR SEGMENT NAME
07800H	CGROUP
	CODE\CODE\ROOT
	CODE\CODE\PASS1
	CODE\CODE\PASS2
	CODE\CODE\PIC_PASS2
	CODE\CODE\FASTLOAD
10340H	DGROUP
	CONST\CONST\ROOT
	DATA\DATA\ROOT
	STACK\STACK\

Figure J-3. LOC86 Listing for Program with Overlays

Example 4: Linking and Locating Programs with Overlays Without OVERLAY Control

It is harder to produce overlay modules without using the OVERLAY control. However, sometimes it is necessary to build programs in this way, for example, building a program for running under an operating system that does not support overlay modules contained in the same file as the root module.

But regardless of the reason, building overlays in this fashion places an extra burden on the programmer. He must do some of the work that would be left to LINK86 (and LOC86) if he were to use the OVERLAY control. In the following example we prepare a root and two overlay modules in separate files.

First we must compile all modules. Examples of the invocation lines are shown below:

```
-PLM86 :F1:ROOT.SRC SMALL<cr>
```

```
-PLM86 :F1:OV1.SRC SMALL<cr>
```

```
-PLM86 :F1:OV2.SRC SMALL<cr>
```

In the next step we must link the root module to resolve external symbols with a library and to obtain a link map:

```
-LINK86 :F1:ROOT.OBJ,USER.LIB MAP<cr>
```

We will need the link map for locating purposes. The link map, shown in figure J-4, shows the size of each segment in the root. Since the overlays are self-contained except for references to the root, we do not need a link map for them. The PL/M-86 listing files will show the size of each overlay's segments, as illustrated in figure J-5.

Note that the length of the root's code segment and OV1's code segment must fit within 64K. This means that the code for the overlays must be in a part of memory contiguous with the root (to avoid altering the CS register during execution). OV2's CONST and DATA segments are larger than OV1's so that the STACK segment must be placed to leave room for OV2's CONST and DATA segments. If the overlays share the STACK and MEMORY segments with the root, they must be located at the same address.

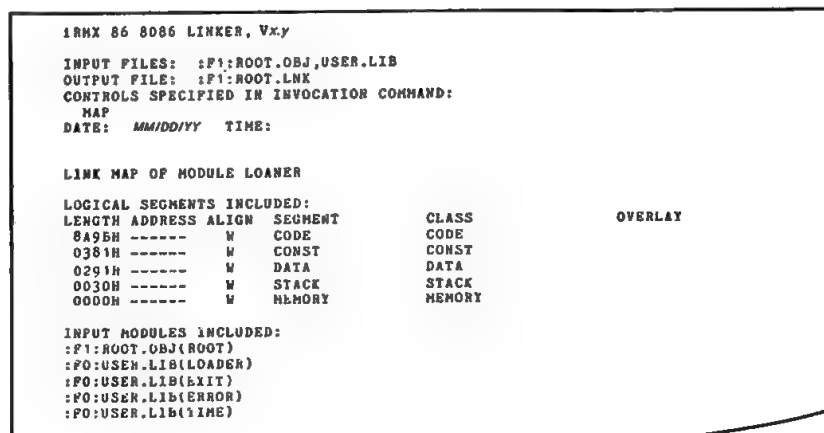


Figure J-4. LINK86 Map for Root File

```

                                OV1's segment size information
MODULE INFORMATION:
  CODE AREA SIZE      = 7531H   30B010   this is the CODE segment
  CONSTANT AREA SIZE  = 0081H   129D    this is the CONST segment
  VARIABLE AREA SIZE  = 0181H   385D    this is the DATA segment
  MAXIMUM STACK SIZE  = 0040H   64D     this is the STACK segment
  918 LINES READ
  0 PROGRAM ERROR(S)
END OF PL/M-86 COMPILATION

```

```

                                OV2's segment size information
MODULE INFORMATION
  CODE AREA SIZE      = 1B9AH   7066D   this is the CODE segment
  CONSTANT AREA SIZE  = 0101H   257D    this is the CONST segment
  VARIABLE AREA SIZE  = 0454H   1108D    this is the DATA segment
  MAXIMUM STACK SIZE  = 0067H   103D    this is the STACK segment
  918 LINES READ
  0 PROGRAM ERROR(S)
END OF PL/M-86 COMPILATION

```

Figure J-5. Module Information for Overlays

After computing the required location for the root's DGROUP and STACK, we can locate the root module. The resulting file will not be executable, but it allows us to resolve references to the root's code and data symbols in the overlays. The following LOC86 invocation will leave room for the overlays' code segments and place the DGROUP in the first unused memory location. (In the command line below, the DS register is initialized to 0FFCEH, and the CS register is initialized to 0.) The STACK and MEMORY segments will be located above OV2's DATA segment:

```

-LOC86 :F1:ROOT.LNK &<cr>
**ADDRESSES(GROUPS(GGROUP(0H),DGROUP(0FFCEH)), &<cr>
**          SEGMENTS(CODE(0H),CONST(0FFCEH),STACK(10B34H))) &<cr>
**ORDER(SEGMENTS(CODE,CONST,DATA,STACK,MEMORY)) &<cr>
**SEGSIZE(STACK(400H)) &<cr>

```

Once the root is located, we can use it to resolve external references in the overlay modules. The overlay modules cannot call each other, since only one is resident in memory at any time. The link commands are shown below. The NOPUBLICS with the EXCEPT control is used to avoid conflicts when we use the located overlays to resolve external references in the root:

```

-LINK86 :F1:OV1.OBJ,PUBLICSONLY(:F1:ROOT) &<cr>
**NOPUBLICS EXCEPT(OV1CODE, OV1DATA)&<cr>

-LINK86 :F1:OV2.OBJ,PUBLICSONLY(:F1:ROOT) &<cr>
**NOPUBLICS EXCEPT(OV2CODE, OV2DATA)&<cr>

```

The PUBLICSONLY control resolves references to public symbols contained in the root.

After the overlays have been linked, they must be located. The code and data segments must be placed in the memory locations that were reserved when we first located the root. In this case the STACK and MEMORY segments must be the same for the overlays and the root:

```
-LOC86 :F1:OV1.LNK &<cr>
**ADDRESSES(GROUPS(CGROUP(0H), DGROUP(0FFCEH)), &<cr>
**      SEGMENTS(CODE(8A9CH), CONST(105E0H), STACK(10B34H))) &<cr>
**ORDER(SEGMENTS(CODE, CONST, STACK, MEMORY)) &<cr>
**SEGSIZE(STACK(100H))<cr>

-LOC86 :F1:OV2.LNK &<cr>
**ADDRESSES(GROUPS(CGROUP(0H), DGROUP(0FFCEH)), &<cr>
**      SEGMENTS(CODE(8A9CH), CONST(105E0H), STACK(10B34H))) &<cr>
**ORDER(SEGMENTS(CODE, CONST, DATA, STACK, MEMORY)) &<cr>
**SEGSIZE(STACK(100H))<cr>
```

The CGROUP and DGROUP base address must be specified in order to compute offset information. The final base address assigned to DGROUP by LOC86 will be rounded down to 0FFC0H.

Once the overlays are located, the root is linked and located into an executable form. The PUBLICONLY control will resolve references to symbols in the overlay modules. Other than the addition of this input control, the LINK86 and LOC86 command must be identical to those used previously:

```
-LINK86 :F1:ROOT.OBJ, USER:LIB, &<cr>
**PUBLICONLY(:F1:OV1, :F1:OV2)<cr>

-LOC86 :F1:ROOT.LNK &<cr>
**ADDRESSES(GROUPS(CGROUP(0H), DGROUP(0FFCEH)), &<cr>
**      SEGMENTS(CODE(0H), CONST(0FFCEH), STACK(10B34H))) &<cr>
**ORDER(SEGMENTS(CODE, CONST, DATA, STACK, MEMORY)) &<cr>
**SEGSIZE(STACK(100H))<cr>
```

The executable forms of the root and its overlay files are contained in :F1:ROOT, :F1:OV1, and :F1:OV2. Figure J-6 shows the resulting layout of memory.

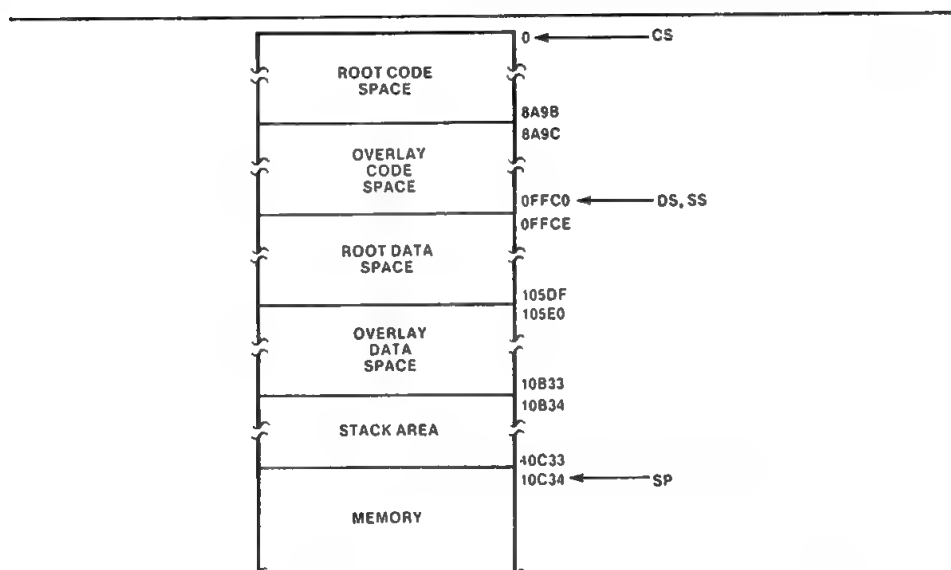


Figure J-6. Memory Organization for Example 4

121616-10

Invocation Examples

The following foldout pages contain examples of the iAPX 86,88 Family utility controls and commands. The examples may be used in conjunction with syntax specifications given:

- In Chapter 2 for LINK86
- In Chapter 3 for CREF86
- In Chapter 4 for LIB86
- In Chapter 5 for LOC86

When using the directions in these chapters, fold out the page in this appendix containing examples of the command or control in which you are interested.

The following is a sample iRMX 86 OH86 invocation:

```
OH86 :FD0:FINALPROGRAM to :FD0:FINISH.HEX<cr>
```

Comments

This example defines two public symbols, VARONE and VARTWO, with absolute addresses 50H and 2000H, respectively.

In this example, the root file is RTFILE, and LIB1 and LIB2 are library files.

This line creates an LTL module. The output object file is :FD0:TEST with no extension.

This example specifies default to avoid ambiguity.

Do not use the FASTLOAD control when producing code for an iRMX environment.

LINES is the default, so it need not be specified.

This MEMPOOL example will increase the minimum dynamic memory requirements by 20H bytes, and by default the maximum size will increase, if necessary, to equal the minimum.

The minimum dynamic memory requirement is 100H. The maximum dynamic memory requirement is 300H.

Control

ASSIGN

ASSUMEROOT

BIND/NOBIND

COMMENTS/NOCOMMENTS

FASTLOAD/NOFASTLOAD

INITCODE

LINES/NOLINES

MAP/NOMAP

MEMPOOL

Examples

```
-LINK86 FILE1, FILE2, FILE3 &<cr>
**ASSIGN (VARONE(50H), &<cr>
**VARTWO(2000))<cr>

-LINK86 ROOT1.OBJ, ROOT2.OBJ &<cr>
**LIB1, LIB2 TO RTFILE OVERLAY<cr>

-LINK86 OV11.OBJ, OV12.OBJ &<cr>
**LIB1, LIB2 TO OV1.OBJ &<cr>
**OVERLAY ASSUMEROOT(RTFILE)<cr>

-LINK86 OV21.OBJ, OV22.OBJ &<cr>
**LIB1, LIB2 TO OV2.OBJ &<cr>
**OVERLAY ASSUMEROOT(RTFILE)<cr>

-LINK86 RTFILE, OV1.OBJ, &<cr>
**OV2.OBJ TO PROG<cr>

-LINK86 :FD0:TEST.OBJ &<cr>
**PROG:LIBRARY/USER.LIB &<cr>
**BIND PRINT<cr>

-LINK86 :SYSTEM:GENERAL.OBJ &<cr>
**NOBIND<cr>

-LINK86 :F1:SOURCE.OBJ &<cr>
**NOCOMMENTS<cr>

-LINK86 :WD0:SYSTEM/PROG.OBJ &<cr>
**TO :WD0:SYSTEM/TEMP.TST &<cr>
**COMMENTS<cr>

-LINK86 PROG.OBJ, LIB1, LIB2 &<cr>
**BIND FASTLOAD<cr>

-LINK86 :PROG:MYPROG INITCODE<cr>

-LINK86 :F1:TEST/RN.OBJ NOLINES<cr>

-LINK86 :F1:TEST/RN.OBJ LINES<cr>

-LINK86 :F1:TESTER.OBJ MAP<cr>

-LINK86 :FD1:MAIN.OBJ &<cr>
**PROG:USER.OBJ &<cr>
**PUBLICONLY(:FD0:8087.LOC) &<cr>
**NOMAP<cr>

-LINK86 :FD0:TEST.OBJ &<cr>
**USER.LIB, PASCAL.LIB BIND &<cr>
**MEMPOOL(+20H)<cr>

-LINK86 :WD0:USER/TEST.OBJ &<cr>
**MEMPOOL(100H, +200H) BIND<cr>
```

Comments	Control	Examples
The LINK86 output module will have the name specified in parentheses in the control.	NAME	<pre>-LINK86 :USER:TOM.LBJ, &<cr> **/SYS.LIB NAME &<cr> **(THIS IS A VERY LONG MODULE@NAME.)<cr> -LINK86 :F1:SRC1.OBJ, &<cr> **F1:SRC2.OBJ, :WD0:USER.LIB &<cr> **NAME(TEST 5.5)<cr></pre>
This example removes all debug and public records from the object file.	OBJECTCONTROLS	<pre>-LINK86 :F1:FINAL, &<cr> **F1:USER.LIB, SYS:MP &<cr> **OBJECTCONTROLS(PURGE)<cr> -LINK86 :PROG:PASCL1.OBJ &<cr> **OBJECTCONTROLS(PURGE, &<cr> **NOPUBLICS EXCEPT(START, &<cr> **DATA1, DATA2))<cr></pre>
The EXCEPT in the NOPUBLICS overrides the PURGE.	ORDER	<pre>-LINK86 :F1:PLMPRG.OBJ, &<cr> **PLM.LIB, URXSML.LIB, &<cr> **USER.LIB ORDER(DGROUP &<cr> **(SEG1, SEG2\CLASS1, &<cr> **SEG2\CLASS1\OVERLAY1), &<cr> **CGROUP(CSEG1, CSEG2, &<cr> **CSEG3))<cr></pre>
This use of ORDER specifies the order of segments for two groups.	OVERLAY/NOOVERLAY	<pre>-LINK86 FILE1, FILE2, FILE3 &<cr> **TO :FD0:OV1.LNK &<cr> **OVERLAY(OVERLAY1)<cr> -LINK86 FILE4, FILE5, FILE6 &<cr> **TO :FD0:OV2.LNK &<cr> **OVERLAY(OVERLAY2)<cr> -LINK86 FILE7, FILE8, FILE9 &<cr> **TO :FD0:ROOT.LNK OVERLAY &<cr> ** (ROOT)<cr> -LINK86 :FD0:ROOT.LNK, &<cr> ** :FD0:OV1.LNK, :FD0:OV2.LNK<cr></pre>
This example will create an overlay record. The name of the overlay will be OVERLAY1.	PRINT/NOPRINT	<pre>-LINK86 :FD0:USER/PROG.OBJ &<cr> **TO :FD0:USER/TEMP1.TST &<cr> **PRINT<cr> -LINK86 :F1:PROG.OBJ<cr> -LINK86 :PROG:PROG.OBJ, &<cr> ** :PROG:USER.LIB PRINT &<cr> ** (:F1:THE.MAP)<cr></pre>
First the constituent files must be linked to form overlays.	PRINT CONTROLS	<pre>-LINK86 :WD0:TEMP.OBJ BIND &<cr> **PRINTCONTROLS(NOLINES, &<cr> **NOCOMMENTS, NOSYMBOLS)<cr> -LINK86 :WD0:PASCL1.OBJ &<cr> **PRINTCONTROLS(PURGE)<cr></pre>
The print file is :FD0:USER/TEMP1.MP1.		
The print file is :F1:PROG.MP1.		
The print file is :F1:THE.MAP.		
This example removes information about line numbers, local symbols, and comments from the print file.		
This statement removes all but the segment information and error messages from the print file.		

Comments

Public information concerning only DATA1, DATA2, LABEL3, and PROC4 is placed in the object file and print file.

All public symbol information will be included in the print file and output file.

This example will produce a file containing only the absolute public symbol records from :F1:8087.LOC. The object file will be :F1:8087.LNK.

This will resolve the references in ROOT.OBJ to absolute public symbols in the separately linked and located overlays OV1 and OV2.

This produces an object file containing no debug or public information.

This confirms that the line and symbol information should be kept in the print file.

This will change the translator-assigned name CGROUP to THE@CODE. A subsequent linkage would not combine THE@CODE with a group named CGROUP.

This changes the name of the CODE group to CGROUP.

This tells the loader that 15FFH bytes of code is the minimum requirement for MEMORY. The new maximum size of MEMORY is 35FFH.

This increments MEMORY's minimum size by 1FFH (7951D) bytes. The maximum size of MEMORY is equal to the old minimum size plus 3FEH (15902D).

The local symbol records will be included in the object file.

PURGE is a shorthand for NOSYMBOLS, NOCOMMENTS, NOPUBLICS, NOTYPE, and NOLINES.

SYMBOLCOLUMNS has no effect, since BIND was not specified.

The symbol table will be printed on a line printer.

LIBMOD will retain its type information.

Control

PUBLICS/NOPUBLICS

PUBLICSONLY

PURGE/NOPURGE

RENAMEGROUPS

SEGSIZE

SYMBOLS/NOSYMBOLS

SYMBOLCOLUMNS

TYPE/NOTYPE

Examples

```
-LINK86 :F1:TEST.OBJ &<cr>
**USER.LIB NOPUBLICS EXGEAT &<cr>
** (DATA1, DATA2, LABEL3, PROC4) <cr>

-LINK86 :F2:TEMP.OBJ &<cr>
**URXSML.LIB, USER.LIB PUBLICS<cr>

-LINK86 PUBLICSONLY &<cr>
** (:F1:8087.LOC) <cr>

-LINK86 :F1:ROOT.OBJ &<cr>
**PUBLICSONLY (:F1:OV1, :F1:OV2) <cr>

-LINK86 :F1:INDEX.OBJ PURGE<cr>

-LINK86 :WD0:FINAL.OBJ &<cr>
**PRINTCONTROLS (NOPURGE) <cr>

-LINK86 :F1:PLMPRG.OBJ &<cr>
**RENAMEGROUPS (CGROUP TO <cr>
**THE@CODE) <cr>

-LINK86 :WD1:ASMPRG.OBJ &<cr>
**RENAMEGROUPS (CODE TO CGROUP) <cr>

-LINK86 :FD1:GEORGE.OBJ &<cr>
**USER.LIB, SYSTEM.LIB :IN1 &<cr>
**SEGSIZE (MEMORY (15FFH, &<cr>
**+2000H)) <cr>

-LINK86 :WD0:PROJECT.OBJ &<cr>
**WD0:REST.LIB SEGSIZE &<cr>
** (MEMORY (+1FF, +1FF)) <cr>

-LINK86 :F1:TEMP.OBJ &<cr>
**F1:USER.LIB NOSYMBOLS &<cr>
**SYMBOLS <cr>

-LINK86 :FD0:TEST.OBJ &<cr>
**FD1:SYSTEM/USER.LIB &<cr>
**PURGE <cr>

-LINK86 :F1:TEST.OBJ &<cr>
**SYMBOLCOLUMNS (1) <cr>

-LINK86 :F1:ROOT.LNK &<cr>
** (:F1:OV1.LNK, :F1:OV2.LNK) &<cr>
**PUBLICSONLY :WD :8087 &<cr>
**SYMBOLCOLUMNS (4) BIND &<cr>
**PRINT (:LP:) <cr>

-LINK86 :F1:LIBMOD.OBJ TYPE<cr>
```

Comments

The cross-reference listing will have 35 lines on each page.

The cross-reference listing will be 100 characters wide maximum per page.

The pathname of the print file will be :FX1:MYFILE.

The message in the TITLE control must be placed on one line. If the message contains special characters, it must be enclosed in single quotes (').

Control

PAGELNGTH

PAGEWIDTH

PRINT

TITLE

Examples

```
-CREF86 :F1:FILE1 FILE.LIB &<cr>
**PAGELENGTH(35)<cr>

-CREF86 :PROG:PROGRAM, &<cr>
** :PROG:LIB(MOD1) &<cr>
**PAGEWIDTH(100)<cr>

-CREF86 :FX1:FILES(MODULE1,MODULE2) &<cr>
**PRINT(:FX1:MYFILE)<cr>

-CREF86 MYPROG,HISPROG,HERPROG &<cr>
**MYLIB,HISLIB,HERLIB TITLE &<cr>
**('A CROSS-REFERENCE') &<cr>
**PAGEWIDTH(105)<cr>
```


Comments	Control	Examples
Three object files are added to the USER.LIB.	ADD	<pre>*ADD :F1:SIN :F1:COS,:F1:TAN TO &<cr> ** :F0:USER.LIB<cr></pre>
Three modules from the library LIB.ABC are added to :FD0:PROJ.TOM.		<pre>*ADD :FD0:LIB.ABC(MOD1, MOD2, MOD3) &<cr> **TO :FD0:PROJ.TOM<cr></pre>
This command will produce an empty library file called TOMS.LIB.	CREATE	<pre>*CREATE :W00:SYSTEM/TOMS.LIB<cr> *CREATE :F0:USER.LIB<cr></pre>
Four modules are deleted from the library USER.LIB.	DELETE	<pre>*DELETE :FX1:USER.LIB(TEMP1, &<cr> **TEMP3, TEN TMP,TEST?)<cr> *DELETE :F0:TO.LIB(FLOPPY, CRT, &<cr> **PAPER TAPE)<cr></pre>
	EXIT	<pre>*EXIT<cr></pre>
	LIST	<pre>*LIST USER.LIB<cr> USER.LIB TEMP TEST EXEC MAIN LOOP *LIST USER.LIB(TEMP,TEST)<cr> USER.LIB TEMP TEST *LIST USER.LIB,TEMP.LIB<cr> USER.LIB TEMP TEST EXEC MAIN LOOP TEMP.LIB MODULE1 MODULE3 MODULE4TC</pre>

Comments

If SEG1 is byte alignable, it will be located at 15FFH.
If SEG2 is byte or word alignable, it will be at 4F5AH.

Address assignment of groups, segments, and
classes can be in any order, as long as addresses do
not conflict with existing absolute addresses.

A long jump to GO will be placed at location
0FFFF0H.

The initialization code is placed at address 32768
decimal (8000H).

No initialization code will be produced.

LINES is the default, so it need not be specified.

This statement removes all debug records from the
object file, but keeps the information in the print file.

NOPUBLICS is implied by PURGE, but PUBLICS
overrides it.

SEG@A of CLASS1 will be the first relocatable
segment located. SEG@B will be next, followed
immediately by any other segments contained within
CLASS1. The extra segments in CLASS1 (and all of
the segments in CLASS2) are located in the order in
which they are encountered. Finally, the list in the
SEGMENTS subcontrol is handled.

Control

ADDRESS

BOOTSTRAP

COMMENTS/NOCOMMENTS

INITCODE/NOINITCODE

LINES/NOLINES

MAP/NOMAP

NAME

OBJECTCONTROLS

ORDER

Examples

-LOC86 :FD1:COME.LNK TO WENT &<cr>
**ADDRESSES(SEGMENTS(SEG1\CLASS2), &<cr>
**OVERLAY3(15FFH), SEG2(4F5AH))<cr>

-LOC86 :F1:IST.LNK TO LOCIST.RDY &<cr>
**ADDRESSES(SEGMENTS(SEG1\CLASS1), &<cr>
** (23H)), GROUPS(CGROU(00H), DGROU &<cr>
** (10000H)), CLASSES(MEMORY(15000H)))<cr>

-LOC86 :WD1:USER/TEST.LNK &<cr>
**START(GO) BOOTSTRAP<cr>

-LOC86 :F1:SOURCE.LNK NOCOMMENTS<cr>

-LOC86 :F1:TEMP.LNK COMMENTS<cr>

-LOC86 :FX1:PROGDIR/FORK.LNK &<cr>
**INITCODE(32768)<cr>

-LOC86 :F1:TEST.LNK NOINITCODE<cr>

-LOC86 :F1:RUN.LNK NOLINES<cr>

-LOC86 :F1:TEST.LNK<cr>

-LOC86 :F1:TESTER.LNK MAP<cr>

-LOC86 :FD0:GONE.LNK TO &<cr>
** :FD0:HERMAF.OVY NOMAP<cr>

-LOC86 :F4:SHORT.LNK NAME &<cr>
** (THIS IS A VERY LONG MODULE)<cr>

-LOC86 :F1:UPWARD.LNK &<cr>
**OBJECTCONTROLS (NOLINES, &<cr>
**NOCOMMENTS, NOSYMBOLS)<cr>

-LOC86 :F3:PASCAL1.LNK &<cr>
**OBJECTCONTROLS (PURGE,PUBLICS)<cr>

-LOC86 :F0:SPCSEQ.LNK ORDER &<cr>
** (CLASSES(CLASS1(SEG@A, SEG@B), &<cr>
** CLASS2), SEGMENTS(SEG1\CLASS1), &<cr>
** \OVERLAY1, SEG22, SEG10\CLASS5))<cr>

Comments
The print file is :F1:TEMP1.MP2.
The print file is :FDD0:INTERRUPT.MP2.
The print file is :F1:MAP.
Information about line numbers is removed from the print file.
All but the segment information is removed from the print file.
No public information is included in the output files (:USER:PRIVATE.MP2 and :USER:PRIVATE).
All public information will be included in both the print file and output file.
The object file contains no public or debug information, and the symbol table does not appear in the print file.
The line and symbol information will be kept in the print file.
This control reserves the high-order 64K of memory.
A 200H and a 100H section of memory at the top and bottom of memory are reserved.
The size of segment MEMORY will be increased by 2000 bytes.
The size of segment MYSEG will be decreased by 511 bytes.
The new segment size for XENDA is 7770 bytes.
:FD1:AUTO will start at IGNITION.
:PROG:HALTS will start at location 200H.
This statement will include the local symbol records in the object file and the symbol information in the print file.
PURGE is a shorthand for NOSYMBOLS, NOCOMMENTS, NOPUBLICS, and NOLINES.
The symbol table will be printed on a line printer. A line printer line can hold a four-column symbol table.

Control
PRINT/NOPRINT
PRINTCONTROLS
PUBLICS/NOPUBLICS
PURGE/NOPURGE
RESERVE
SEGSIZE
START
SYMBOLS/NOSYMBOLS
SYMBOLCOLUMNS

Examples
<pre>-LOC86 :F1:PROG.LNK TO &<cr> ** :F1:TEMP1.MP2 PRINT<cr></pre>
<pre>-LOC86 :FDD0:INTERRUPT.LNK<cr> -LOC86 :WDD:PROG.LNK PRINT(:F1:MAP)<cr></pre>
<pre>-LOC86 :F1:LINEAR.LNK &<cr> **PRINTCONTROLS(NOLINES)<cr> -LOC86 :WD1:DIR1/SBDIR/PR.LNK &<cr> **PRINTCONTROLS(PURGE)<cr></pre>
<pre>-LOC86 :USER:PRIVATE.LNK NOPUBLICS<cr> -LOC86 :PROG:TEXT.LNK &<cr> **NOPUBLICS PUBLICS<cr></pre>
<pre>-LOC86 :F3:PROJ5.LNK PURGE<cr> -LOC86 :FX0:B0209.LNK PURGE &<cr> **PRINTCONTROLS (NOPURGE)<cr></pre>
<pre>-LOC86 :F1:LOWMEM.LNK RESERVE &<cr> ** (0F0000H TO 0FFFFFFH)<cr> -LOC86 :F2:HUGOS.LNK RESERVE &<cr> ** (00H TO 0200H, 0FFF00H TO 0FFFFFFH)<cr></pre>
<pre>-LOC86 :WD1:DIREC/GROW.LNK SEGSIZE &<cr> ** (MEMORY(+2000))<cr> -LOC86 SEGPROB.LNK SEGSIZE &<cr> ** (MYSEG(-1FFH))<cr></pre>
<pre>-LOC86 :F1:RPLACE.LNK SEGSIZE &<cr> ** (XENDA(7770))<cr></pre>
<pre>-LOC86 :FD1:AUTO.LNK START(IGNITION)<cr> -LOC86 :PROG:HALTS.LNK START &<cr> ** (00H, 200H)<cr></pre>
<pre>-LOC86 GESHTA.LNK SYMBOLS<cr></pre>
<pre>-LOC86 :F3:TEST.LNK PURGE<cr></pre>
<pre>-LOC86 :F3:DIR2/TEST.LNK &<cr> **SYMBOLCOLUMNS(1)<cr> -LOC86 :USER:LINKED.LNK &<cr> **SYMBOLCOLUMNS(4) PRINT(:LP)<cr></pre>



- absolute object file formats, A-1
- absolute object modules, 1-2
- AD, 5-3
- address,
 - in ADDRESSES control, 5-3
 - in ASSIGN control, 2-4
 - in INITCODE control, 5-6
 - in RESERVE control, 5-16
- ADDRESSES, 5-3
- addressing,
 - A, 4-2
 - absolute, 1-4, 2-4
 - ADD, 4-2
 - 8086, 1-5
 - relative, 1-4
- alignment,
 - boundaries, 1-8
 - of segments, 1-7
- AR, 2-5
- AS, 2-4
- ASSIGN, 2-4
- available memory, effect of, C-1
- BI, 2-6
- BIND, 2-6
- BOOTSTRAP, 5-4
- bound modules (*see LTL modules*)
- BS, 5-4
- C, 4-3
- class, 8086, 1-9
- CLASSES, 5-3, 5-11
- class name,
 - in ADDRESSES control, 5-3
 - in ORDER control,
 - LINK86, 2-15
 - LOC86, 5-11
 - in SEGSIZE control,
 - LINK86, 2-23
 - LOC86, 5-17
- CM, 2-7
- COMMENTS,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86 control, 2-7
 - LOC86 control, 5-5
- control summary,
 - CREF86, E-1
 - LIB86, F-1
 - LINK86, D-1
 - LOC86, G-1
- CREATE, 4-3
- CREF86,
 - controls,
 - PAGELNGTH, 3-3
 - PAGEWIDTH, 3-4
 - PRINT, 3-5
 - TITLE, 3-6
 - control summary, 3-2
 - error messages, E-1
 - in development process, 1-1
 - input, 3-1
 - invocation, 3-2
 - iRMX 86 invocation examples, J-19/J-20 |
 - output, 3-1
 - print file, 3-7
 - Series III invocation examples, I-23/I-24
 - use of libraries, 1-3
- cross-reference listing, 3-7
- CS, 5-3, 5-11
- D, 4-3
- data records, 8086, A-3
- debug records,
 - LINK86, 2-21
 - LOC86, 5-15
- DELETE, 4-3
- E, 4-4
- ENDREC, A-7
- error messages,
 - CREF86, E-1
 - LIB86, F-1
 - LINK86, D-2
 - LOC86, G-2
 - OH86, H-1
- examples, iRMX 86
 - invocation,
 - CREF86, J-19/J-20
 - LIB86, J-21/J-22
 - LINK86, J-13/J-14
 - LOC86, J-23/J-24
 - OH86, J-12
 - program development
 - CREF86, J-3
 - LIB86, J-5
 - LINK86, J-6, J-9
 - LOC86, J-10
- examples, Series III,
 - invocation,
 - CREF86, I-23/I-24
 - LIB86, I-25/I-26
 - LINK86, I-17/I-18
 - LOC86, I-27/I-28
 - OH86, I-15
 - program development,
 - CREF86, I-6
 - LIB86, I-7
 - LINK86, I-2, I-5, I-8, I-11, I-14
 - LOC86, I-4, I-5, I-8, I-12

- EXIT, 4-4
- external references,
 - cross-reference listing, 3-7
 - definition of, 1-2
 - resolution of, 1-3
- FASTLOAD, 2-8
- FL, 2-8
- GR, 5-3
- group,
 - addressing, 1-9
 - 8086, 1-9
- group map, 2-27
- group name,
 - in ADDRESSES control, 5-3
 - in LINK86 ORDER control, 2-15
 - in LINK86 RENAMEGROUP control, 2-22
- GROUPS, 5-3
- hexadecimal-decimal conversion, B-1
- hexadecimal object file format,
 - conversion to, 6-1
 - records of,
 - data, A-13
 - end of file, A-15
 - extended address, A-12
 - start address, A-14
- IC,
 - LINK86, 2-9
 - LOC86, 5-6
- INITCODE,
 - LINK86, 2-9
 - LOC86, 5-6
- initialization code,
 - LINK86, 2-9
 - LOC86, 5-6
- input list control, 2-20
- iRMX 86 information,
 - continuation-line characters, J-2
 - environmental conditions, J-1
 - examples, J-12
 - file-naming conventions, J-2
 - generation of code, J-3
 - invocation, J-1
 - memory requirements, J-1
 - program development examples, J-3
 - prompts, J-2
 - related publications, J-2
- L, 4-6
- LHEADR, A-6
- LI,
 - LINK86, 2-10
 - LOC86, 5-7
- LIB86,
 - commands,
 - ADD, 4-2
 - CREATE, 4-3
 - DELETE, 4-4
 - EXIT, 4-5
 - LIST, 4-6
 - command summary, 4-1, F-1
 - error messages, F-1
 - in development process, 1-1
 - input, 4-1
 - invocation, 4-1
 - iRMX 86 invocation examples, J-21/J-22
 - Series III invocation examples, 1-25/1-26
- librarian (see *LIB86*)
- libraries,
 - adding to, 4-2
 - creating, 4-3
 - deleting from, 4-4
 - listing contents of, 4-6
 - use of by CREF86, 1-3
 - use of by LINK86, 1-3
- line number control,
 - LINK86, 2-10
 - LOC86, 5-7
- LINES,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-10
 - LOC86, 5-7
- link map, 2-27
- linkage (see *LINK86*)
- LINK86,
 - and LOC86, 1-4
 - controls,
 - ASSIGN, 2-4
 - ASSUMEROOT, 2-5
 - BIND, 2-6
 - COMMENTS, 2-7
 - FASTLOAD, 2-8
 - INITCODE, 2-9
 - LINES, 2-10
 - MAP, 2-11
 - MEMPOOL, 2-12
 - NAME, 2-13
 - NOBIND, 2-6
 - NOCOMMENTS, 2-7
 - NOFASTLOAD, 2-8
 - NOLINES, 2-10
 - NOMAP, 2-11
 - NOOVERLAY, 2-16
 - NOPRINT, 2-17
 - NOPUBLICS, 2-19
 - NOPURGE, 2-21
 - NOSYMBOLS, 2-24
 - NOTYPE, 2-26
 - OBJECTCONTROLS, 2-14
 - ORDER, 2-15
 - OVERLAY, 2-16
 - PRINT, 2-17
 - PRINTCONTROLS, 5-12
 - PUBLICS, 2-19
 - PUBLICSONLY, 2-20
 - PURGE, 2-21
 - RENAMEGROUPS, 2-22
 - SEGSIZE, 2-23
 - SYMBOLCOLUMNS, 2-25
 - SYMBOLS, 2-24
 - TYPE, 2-26

- control summary, 2-2, D-1
- error messages, D-2
- in development process, 1-1
- input, 1-4, 2-1, 2-20
- invocation, 2-1
- | iRMX 86 invocation examples, J-13/J-14
- output, 1-4, 2-1
- print file, 2-27
- segment combination, 1-7
- Series III invocation examples, 1-17/1-18
- use of libraries, 1-3
- LIST, 4-6
- load-time-locatable module (*see LTL module*)
- location (*see LOC86*)
- location algorithm,
 - for modules with overlays, 5-25
 - for segments, 5-24
- LOC86,
 - and LINK86, 1-4
 - controls,
 - ADDRESSES, 5-3
 - BOOTSTRAP, 5-4
 - COMMENTS, 5-5
 - INITCODE, 5-6
 - LINES, 5-7
 - MAP, 5-8
 - NAME, 5-9
 - NOCOMMENTS, 5-5
 - NOINITCODE, 5-6
 - NOLINES, 5-7
 - NOMAP, 5-8
 - NOPRINT, 5-12
 - NOPUBLICS, 5-14
 - NOPURGE, 5-15
 - NOSYMBOLS, 5-19
 - OBJECTCONTROLS, 5-10
 - ORDER, 5-11
 - PRINT, 5-12
 - PRINTCONTROLS, 5-13
 - PUBLICS, 5-14
 - PURGE, 5-15
 - RESERVE, 5-16
 - SEGSIZE, 5-17
 - START, 5-18
 - SYMBOLS, 5-19
 - SYMBOLCOLUMNS, 5-19
 - control summary, 5-2, G-1
 - error messages, G-2
 - in development process, 1-1
 - input, 1-4, 5-1
 - invocation, 2-1
 - | iRMX 86 invocation examples, J-23/J-24
 - output, 1-4, 5-1
 - print file, 5-21
 - Series III invocation examples, 1-27/1-28
- LTL controls,
 - BIND, 2-6
 - FASTLOAD, 2-8
 - MEMPOOL, 2-12
 - ORDER, 2-15
 - PRINTCONTROLS, 2-18
 - SEGSIZE, 2-23
 - SYMBOLCOLUMNS, 2-25
- LTL modules, 1-2, 1-4, 1-10
- MA,
 - LINK86, 2-11
 - LOC86, 5-8
- MAP,
 - LINK86, 2-11
 - LOC86, 5-8
- maximum-size,
 - in MEMPOOL control, 2-12
 - in SEGSIZE control, 2-23
- memory,
 - configuration with overlays, 1-10
 - 8086, 1-5
- memory map, 5-23
- memory requirements controls,
 - LINK86 MEMPOOL, 2-12
 - SEGSIZE,
 - LINK86, 2-23
 - LOC86, 5-17
- MEMPOOL, 2-12
- minimum-size,
 - in MEMPOOL control, 2-12
 - in SEGSIZE control, 2-23
- MODEND, A-9
- module attributes, A-2
- module identification, A-2
- module name,
 - in LINK86 NAME control, 2-13
 - in LOC86 NAME control, 5-9
- MP, 2-12
- NA,
 - LINK86, 2-13
 - LOC86, 5-9
- NAME,
 - LINK86, 2-13
 - LOC86, 5-9
- naming output module,
 - LINK86, 2-13
 - LOC86, 5-9
- NOBI, 2-6
- NOBIND, 2-6
- NOCM, 2-7
- NOCOMMENTS,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-7
 - LOC86, 5-5
- NOFASTLOAD, 2-8
- NOFL, 2-8
- NOIC, 5-6
- NOINITCODE, 5-6
- NOLI,
 - LINK86, 2-10
 - LOC86, 5-7
- NOLINES,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13

- LINK86, 2-10
- LOC86, 5-7
- NOMA,
 - LINK86, 2-11
 - LOC86, 5-8
- NOMAP
 - LINK86, 2-11
 - LOC86, 5-8
- NOOV, 2-16
- NOOVERLAY, 2-16
- NOPL,
 - LINK86, 2-19
 - LOC86, 5-14
- NOPR,
 - LINK86, 2-17
 - LOC86, 5-12
- NOPRINT,
 - LINK86, 2-17
 - LOC86, 5-12
- NOPU,
 - LINK86, 2-21
 - LOC86, 5-15
- NOPUBLICS,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-19
 - LOC86, 5-14
- NOPURGE,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-21
 - LOC86, 5-15
- NOSB,
 - LINK86, 2-24
 - LOC86, 5-19
- NOSYMBOLS,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-24
 - LOC86, 5-19
- NOTY, 2-26
- NOTYPE,
 - in OBJECTCONTROLS, 2-14
 - in PRINTCONTROLS, 2-18
 - LINK86, 2-26
- OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
- object module format, 1-4, A-1
- OC,
 - LINK86, 2-14
 - LOC86, 5-10
- OD,
 - LINK86, 2-15
 - LOC86, 5-11
- offset, 5-18
- OH86,
 - error messages, H-1
 - in development process, 1-1
 - input, 6-1
 - invocation, 6-1
 - iRMX 86 invocation examples, J-12
 - output, 6-1
 - Series III invocation example, I-15
- ORDER,
 - LINK86, 2-15
 - LOC86, 5-11
- OV, 2-16
- OVERLAY, 2-16
- overlay controls,
 - ASSUMEROOT, 2-5
 - OVERLAY, 2-16
- overlay, 8086, 1-10
- overlay name,
 - ADDRESSES, 5-3
 - LINK86 ORDER control, 2-15
 - LINK86 OVERLAY control, 2-16
 - SEGSIZE,
 - LINK86, 2-23
 - LOC86, 5-17
- overlays and location, 5-25
- OVLDEF, A-7
- PAGELength, 3-3
- PAGEWIDTH, 3-4
- paragraph, 5-18
- pathname,
 - in ASSUMEROOT control, 2-5
 - in LIB86 commands, 4-1
 - in PRINT control,
 - LINK86, 2-17
 - LOC86, 5-12
 - in PUBLICSONLY control, 2-20
- PC,
 - LINK86, 2-18
 - LOC86, 5-13
- PEDATA, A-10
- performance-memory relationship, C-1
- PIC, 1-10
- PIDATA, A-10
- PL,
 - CREF86, 3-3
 - LINK86, 2-19
 - LOC86, 5-14
- PO, 2-20
- position-independent code (see PIC)
- PR,
 - CREF86, 3-5
 - LINK86, 2-17
 - LOC86, 5-12
- PRINT,
 - CREF86, 3-5
 - LINK86, 2-17
 - LOC86, 5-12
- PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13

- print file,
 - controls,
 - CREF86, 3-2
 - LINK86, 2-18
 - LOC86, 5-13
 - CREF86,
 - cross-reference information, 3-8
 - header, 3-7
 - module list, 3-8
 - warnings, 3-7
 - LINK86,
 - error messages, 2-30
 - group map, 2-28
 - header, 2-27
 - link map, 2-11, 2-27
 - symbol table, 2-29
 - LOC86,
 - errors and warnings, 5-24
 - memory map, 5-23
 - symbol table, 5-21
- print file name,
 - LINK86, 2-17
 - LOC86, 5-12
- program development, 1-1
- PU,
 - LINK86, 2-21
 - LOC86, 5-15
- PUBLICS,
 - in LIB86 LIST control, 4-6
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-19
 - LOC86, 5-14
- PUBLICSONLY, 2-20
- public symbol, 1-2, 5-18
- public symbol cross-references, 3-7
- public symbol records,
 - in libraries, 4-6
 - LINK86,
 - PUBLICS/NOPUBLICS, 2-19
 - PUBLICSONLY, 2-20
 - LOC86, 5-14
- PURGE,
 - in OBJECTCONTROLS,
 - LINK86, 2-14
 - LOC86, 5-10
 - in PRINTCONTROLS,
 - LINK86, 2-18
 - LOC86, 5-13
 - LINK86, 2-21
 - LOC86, 5-15
- PW, 3-4
- record formats,
 - end, A-7
 - L-module header, A-6
 - module end, A-9
 - overlay definition, A-7
 - physical enumerated data, A-10
 - physical iterated data, A-10
 - register initialization, A-8
 - R-module header, A-6
 - sample, A-4
 - T-module header, A-5
- record syntax, A-3
- REGINT, A-8
- register initialization, 2-9, 5-6, A-8
- relocatable object module, 1-2
- relocation (see *LOC86*)
- RENAMEGROUPS, 2-22
- RESERVE, 5-16
- RG, 2-22
- RHEADR, A-6
- RS, 5-16
- SAMREC, A-4
- SB,
 - LINK86, 2-24
 - LOC86, 5-19
- SC,
 - LINK86, 2-25
 - LOC86, 5-20
- segment,
 - alignment, 1-7, 2-28
 - combining, 1-8
 - 8086, 1-6, A-2
 - locating, 1-8, 5-24
 - memory, 1-8, 2-23, 5-17
 - ordering,
 - LINK86, 2-15
 - LOC86, 5-11, 5-24
 - stack, 1-8
- segment addressability, A-2
- segment location algorithm,
 - absolute segments, 5-24
 - relocatable segments, 5-25
 - segment ordering, 5-24
- segment map, 2-29
- segment name,
 - in ADDRESS control, 5-3
 - in ORDER control,
 - LINK86, 2-15
 - LOC86, 5-11
 - in SEGSIZE control,
 - LINK86, 2-23
 - LOC86, 5-17
- SEGMENTS, 5-3
- SEGSIZE,
 - LINK86, 2-23
 - LOC86, 5-17
- Series III information,
 - continuation-line characters, 1-1
 - environmental considerations, 1-1
 - ISIS-II conventions, 1-1
 - program development examples, 1-2
 - prompts, 1-1
 - related publications, 1-2
 - software version compatibilities, 1-1
- size, 5-17
- SM, 5-3, 5-11
- SS,
 - LINK86, 2-23
 - LOC86, 5-17
- ST, 5-18
- START, 5-18
- start address, 5-4, 5-18

SYMBOLCOLUMNS,
 LINK86, 2-25
 LOC86, 5-20
SYMBOLS,
 in OBJECTCONTROLS,
 LINK6, 2-14
 LOC86, 5-10
 in PRINTCONTROLS,
 LINK86, 2-18
 LOC86, 5-13
 LINK86, 2-24
 LOC86, 5-19
symbol table,
 LINK86, 2-25, 2-29
 LOC86, 5-20, 5-21

THEADR, A-5
TITLE, 3-6
TT, 3-6
TY, 2-26
TYPE,
 in OBJECTCONTROLS, 2-14
 in PRINTCONTROLS, 2-18
 LINK86 control, 2-26
type checking, 2-26

variable name, 2-4



REQUEST FOR READER'S COMMENTS

Intel's Technical Publications Departments attempt to provide publications that meet the needs of all Intel product users. This form lets you participate directly in the publication process. Your comments will help us correct and improve our publications. Please take a few minutes to respond.

Please restrict your comments to the usability, accuracy, readability, organization, and completeness of this publication. If you have any comments on the product that this publication describes, please contact your Intel representative. If you wish to order publications, contact the Intel Literature Department (see page ii of this manual).

1. Please describe any errors you found in this publication (include page number).

2. Does the publication cover the information you expected or required? Please make suggestions for improvement.

3. Is this the right type of publication for your needs? Is it at the right level? What other types of publications are needed?

4. Did you have any difficulty understanding descriptions or wording? Where?

5. Please rate this publication on a scale of 1 to 5 (5 being the best rating). _____

NAME _____ DATE _____

TITLE _____

COMPANY NAME/DEPARTMENT _____

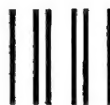
ADDRESS _____

CITY _____ STATE _____ ZIP CODE _____
(COUNTRY)

Please check here if you require a written reply. ☐

WE'D LIKE YOUR COMMENTS ...

This document is one of a series describing Intel products. Your comments on the back of this form will help us produce better manuals. Each reply will be carefully reviewed by the responsible person. All comments and suggestions become the property of Intel Corporation.



BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1040 SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Intel Corporation
Attn: Technical Publications M/S 6-2000
3065 Bowers Avenue
Santa Clara, CA 95051

NO POSTAGE
NECESSARY
IF MAILED
IN U.S.A.

